# HALMA - Dokumentacja Projektu

Antoni Przybylik
PIPR 22Z
GRUPA 101

20 stycznia 2023

# Zadanie

Napisać program grający w Halma. Powinna być możliwość gry:

- dwóch osób ze sobą,
- osoby z komputerem,
- komputera z komputerem.

Program powinien kontrolować poprawność wykonywanych ruchów. Interfejs z użytkownikiem może być tekstowy.

Bardzo istotną częścią zadania jest opracowanie i zaimplementowanie jak najlepszego algorytmu grającego w grę.

Moduł realizujący algorytm gry komputera musi być wydzielony.

# Cel i opis projektu

Celem projektu było napisanie gry Halma w Pythonie, przygotowanie testów jednostkowych i dokumentacji. W tym celu należało wykorzystać umiejętności zdobyte podczas laboratoriów z przedmiotu PIPR.

# Architektura

Projekt jest podzielone na trzy główne moduły:

- Silnik gry.
- Interfejs użytkownika.
- Klasy graczy.

Dodatkowo zawiera jeszcze trzy klasy pomocnicze:

- Silnik UI.
- Interfejs gry dla użytkownika.
- Klasę Game reprezentującą stan całej gry.

W klasie Engine zaimplementowana jest bazowa funkcjonalność programu, między innymi metody odpowedzialne za ustawianie/odczyt pól na planszy, metody pozwalające na sprawdzenie możliwych ruchów z danego pola. W tej klasie jest zapisany stan planszy i numer obecnego ruchu.

W klasie HalmaTui zaimplementowano tekstowy interfejs użytkownika dla gry Halma. Używa on surowych metod do rysowania okienek i tak dalej, z klasy TuiEngine.

Klasy graczy (dziedziczące po Player) implementują metodę make_move(), która odpowiada za wykonywanie ruchu. W przypadku botów ruch jest po prostu wykonywany. Gracze-ludzie (TuiPlayer dziedziczący po Player) wyświetlają dialog box'a przy użyciu interfejsu klasu TuiEngine (tej samej której używa HalmaTui) pytającego gracza o ruch, a potem go wykonują.

Silnik UI znajdujący się w klasie TuiEngine implementuje metody takie jak: rysująca dialog box'a, wyświetlająca splash screen.

Interfejs gry dla użytkownika znajduje się w klasie GameInterface. Celem istnienia tej klasy jest wprowadzenie dodatkowej warstwy abstrakcji pomiędzy UI, a silnik gry. Na przykład ruchy wpisane przez użytkownika nie są sprawdzane i przetwarzane na współrzędne w tych samych funkcjach które wyświetlają UI tylko są przekazywane klasie GameInterface, która ma referencję na obiekt klasy Engine i wykonuje ona ruch. Referencję na obiekt klasy GameInterface ma UI i TuiPlayer, który jest obiektem klasy Player implementującym grę przez użytkownika.

Klasa Game posiada referencję na obiekty klas: GameInterface, Engine i obiekty reprezentujące gracza białego i czarnego (obiekty klas dziedziczących po Player). Ten stan można zapisać do pliku i wczytać go z pliku.

## Wymagania

Żeby móc uruchomić grę w trbie TUI należy mieć zainstalowaną bibliotekę curses.

Interfejs TUI działa tylko w terminalach obsługujących 8-bitowe kolory i pozwalających na zmianę ich wartości[1]. Dodatkowo, jest wymagane żeby okno terminala miało rozmiar co najmniej 40x70[2].

## Instrukcja użycia

W ramach projektu jest zaimplementowany interfejs tekstowy (TUI). Grę w trybie TUI otwieramy uruchamiając plik wykonywalny halma-tui.py.

Komunikacja z programem odbywa się przez dialog boxy. Są ich dwa rodzaje: Pytające o wybór opcji z listy i pytające o inny ciąg znaków.

W dialog box'ie pierwszego rodzaju należy wprowadzić w polu tekstowym numer wybranej opcji i zatwierdzić klawiszem enter.

Inaczej postępujemy z dialog box'em pytającym nas o ruch. W nim należy wprowadzić ruch w formacie „AB-CD"[3], gdzie AB to współrzędne pola z krórego chcemy wykonać ruch, CD to współrzędne pola na które chcemy wykonać ruch. Współrzędne to para literek którymi podpisane są pola z boku planszy.

## Część refleksyjna

Aplikacja ma przejrzystą architekturę z modułami oddzielonymi w sposób przejrzysty bez zagnieżdżania i wstecznych referencji (topologia aplikacji jest acyklicznym grafem skierowanym).

W grze jest możliwość zapisu i wczytania gry, są zaimplementowane dwa boty - losowy i sprytny. Jest zaimplementowane wiele testów. Interfejs użytkownika jest intuicyjny i łatwy w obsłudze, a aplikacja posiada duże możliwości rozszerzenia dzięki enkapsulacji.

Przykładowo, można zaimplementować GUI które komunikowało by się z resztą aplikacji przez taki sam interfejs jak TUI m. Można zaimplementować więcej botów,

[1] Każdy nowoczesny terminal jak urxvt, xfce4-terminal, gnome-terminal powinien spełniać te wymagania. Gra nie działa w xtermie.

[2] Wysokość x Szerokość.

[3] Wielkość liter nie ma znaczenia, nie należy natomiast wprowadzać spacji wewnątrz zapisu.

a nawet możliwość gry zdalnej.

Jest nawet możliwość doimplementowania innej wersji TUI na bazie interfejsu klasy TuiEngine.